KinGUI Programmer Manual

Event description

Most of the user interaction is performed through events issued by the framework to your managers. In this section well enumerate all the possible events you may receive, and how to use the provided information to get the job done. This is a short list:

Event	Description
k_kgmeNOP	
k_kgmeInit	Issued just after the creation of a window -or window overlay- (and before it is displayed for the first time). It is intended to let the user fill any window items that are dynamically set, and also to prepare application dependent buffers, variables, etc.
k_kgmeDone	Issued when a window is about to be destroyed. You receive the event *before* anything is destroyed, so you have access to all the window structures (window items, buffers, etc). It is intended to allow you to save things, but specially to release all the buffers you may hold.
k_kgmeAct	Issued on activation of a window item. This is a very generic event, and usually a specific examination of the *itm* structure is required to take an action.
k_kgmwPkt	Issued when a network packet is received with CR1 field holding your window *wID*. It carries a pointer to the received packet.
k_kgmeMod	Issued on modification of state of a window item, this is as generic as k_kgmeAct, it often is issued in combination with it but should not be confused with each other.
k_kgmeMsg	Issued by user application to user application, this is an 'Interprocess Communication' method based in messages. User just provides a pointer to a data buffer, its size, and a message ID (also user defined).
k_kgmeSelMod	Issued to indicate a change in the selection state of a window item, it also supplies a 'Row' value (and sometimes even a 'Col') for lists to give a clue on what is changing.
k_kgmeLocMenu	Issued upon different strategies, indicated by the 'Func' field: 1 indicates that menu is being composed, and a user manager may remove or add options. 2 just notifies about a user selection, providing the *mID* of the choosed option.
k_kgmeSEv	This is a subscription event: an application registers to receive notifications when specific events happen. To register, you use a registration function.
k_kgmeTmr	This is a timer event: you receive it after a specified time lapse. A timer may be one-time or just repeat periodically, depending on the attributes you set on kgw_TimerSet() call.

Here comes the detailed list with most of the cases:

Event	Description
k_kgmeNOP	
k_kgmeInit	Issued just after the creation of a window -or window overlay- (and before it is displayed for the first time). It is intended to let the user fill any window items that are dynamically set, and also to prepare application dependent buffers, variables, etc. This can be seen as a local window constructor. User often obtains window pointer, and uses *UsrD* pointer to assign some memory buffer (often a structure) with internal variables to keep state information for the duration of the window. In case you forget to free it, it will be done automatically on window destructor -provided you don't have more allocated pointers inside-. Please don't forget to set it back to NULL if you free it on your own destructor (see k_kgmeDone).
k_kgmeDone	Issued when a window is about to be destroyed. You receive the event *before* anything is destroyed, so you have access to all the window structures (window items, buffers, etc). It is intended to allow you to save things, but specially to release all the buffers you may hold. You may also want to send messages to other windows, telling them about your imminent destruction, sending them some data, state, etc. Please don't forget to set freed *UsrD* window pointer to NULL, or the window manager will try to free it again and break the heap.
k_kgmeAct	Issued on activation of a window item. This is a very generic event, and usually a specific examination of the *itm* structure is required to take an action. These are some hints depending on the item type: - t_kwWinI1 (Static Text): Issues this event whenever it is clicked (if it is not disabled and it is visible) - t_kwWinI2 (Edit Text): Issues this event whenever Enter key is pressed (main or keypad) - t_kwWinI5 (Button): Issues this event whenever it is pressed (clicked or space pressed), if it is not disabled and it is visible - t_kwWinI7 (Graphic): Issues this event in case there's a click on the item area (scroll bar is excluded: no event is generated then). User may test graphic cursor (*cIdx* field) to know what variable or task was clicked. For resource-based graphics, it also sets current task (selected by user on the graphic) pointed by *cIdx* (check *Flg.b0* to ensure it was due to this action though). - t_kwWinI9 (Check Box): Issues this event whenever it is clicked (if it is not disabled and it is visible). User should check for b0 in St field to know about its current status. This event is issued *after* status has changed; before that, a k_kgmeMod event was issued, just *before* changing status. - t_kwWinI11 (Radio Button):

Event	Description
	Issues this event when its state goes to 'selected', but you may check also St field for b0 set.
k_kgmwPkt	Issued when a network packet is received with CR1 field holding your window *wID*. It carries a pointer to the received packet (be careful not to modify it as it is not copied but just referenced on the receive queue).
k_kgmeMod	Issued on modification of state of a window item, this is as generic as k_kgmeAct, it often is issued in combination with it but should not be confused with each other. Some hints about its usage: - t_kwWinI2 (Edit Text): Issues this event whenever content is modified by the user interaction (not if you modify it setting its value). - t_kwWinI3 (List Box): Issues this event whenever its selection state changes by the user interaction: toggle selection of a row in multiple select list (in such case, a k_kgmeSelMod event is generated just before this one), change selection on single selection list (in this case, k_kgmeSelMod event will be issued just after this one). Also when cursor is just moved (with keys). It is also issued when the application selects a row using kgw_SetWIi_LBSel(), but this shouldn't happen, and this behavior will probably be removed. - t_kwWinI4 (Combo Box): Issues this event whenever its selection changes by the user interaction. Actually, this event is issued just after changing to the new selection value; a k_kgmeSelMod event is issued just before changing the previous selection value. This event is also issued when user uses the key fast filter (types part of a word to filter visible values), because the filter string (available as 'me' field) has changed. In this case, no k_kgmeSelMod is issued as selection didn't change. - t_kwWinI6 (DateTime): Issues this event after its value is changed (by user interaction, no event is issued if you set it from application). You may check current date/time value ('DT' field). - t_kwWinI9 (Check Box): Issues this event just *before* its state is toggled. Then, a k_kgmeAct event is sent *after* state has changed to the new value. - t_kwWinI11 (Radio Button): Issues this event just *before* changing its state to 'selected'; then it is selected and k_kgmeAct event is issued; finally, the previously selected Radio Button is notified with this event (once its state is set to 'unselected').
k_kgmeMsg	Issued by user application to user application, this is an 'Interprocess Communication' method based in messages. User just provides a pointer to a data buffer, its size, and a message ID (also user

Event	Description
	defined). Care must be taken to ensure that processes don't fill message buffers out of their size. KinGUI framework also uses messages, internally, for some high level operations, but never to a user window manager.
k_kgmeSelMod	Issued to indicate a change in the selection state of a window item, it also supplies a 'Row' value (and sometimes even a 'Col') for lists to give a clue on what is changing. - t_kwWinI3 (List Box): Issues this event when user double-clicks a cell and the list is not editable (so user may take care of the action), or simply clicks on a cell (for multiple select, a k_kgmeMod is also issued, just following this; for a single selection list, this event happens *after* setting the selection state, a k_kgmeMod is issued *before* that); in these cases, 'Col' field is usually valid (65535 indicates invalid: out of range or not applicable). It is also issued when user just moves the selection with some keys, in these cases, 'Col' field is not valid, and 'Row' is the one that is loosing the cursor (as this event is issued *before* changing position); a k_kgmeMod event is issued *after* position has changed. - t_kwWinI4 (Combo Box): Issues this event when user clicks on a value on the list (and only then, keys don't issue this event), changing current selected value. This event happens *after* the new selection value is in effect; *before* that, a k_kgmeMod event is issued so user can react to a loose of selection on a specific value.
k_kgmeLocMenu	Issued upon different strategies, indicated by the 'Func' field: a value of 1 indicates that menu is being composed, and a user manager receiving this event may remove system options, or add its own ones, before it is displayed. A value of 2 just notifies about a user selection, providing the *mID* of the choosed option. User may just return 0 to allow KinGUI framework to perform the default actions for system options. Returning a different value will prevent any action to be performed.
k_kgmeSEv	This is a subscription event: an application registers to receive notifications when specific events happen. To register, you use a registration function; currently, only network monitoring is implemented, by calling king_nwEvent() to register whatever you want to get. It works based on a mask (of the events you want to receive), specific for every module and registration function. For an instance, for the network module, mask bits are as follows: b0=>connection list changes, b1=>connection status changes, b2=>Tx bytes changes, b3=>Rx bytes changes. This event will be issued for other modules in the future as well.
k_kgmeTmr	This is a timer event: you receive it after a specified time lapse. A timer may be one-time or just repeat periodically, depending on the attributes you set on kgw_TimerSet() call. You receive the tID value you set upon timer creation, for your reference.